```
DB_NAME=dabiegorden49
DB_PASSWORD=cug_hostels_bookings
```

Project Structure & Technical Setup

```
/app
 /api          # API routes for backend functionality
 /components     # Reusable UI components
 /lib          # Utility functions and shared code
 /models        # MongoDB schema models
 /public        # Static assets
 /(routes)       # Page routes based on app router
  /page.jsx     # Homepage
  /hostels/page.jsx
  /bookings/page.jsx
  /admin/page.jsx
```

/auth/page.jsx

2. Core Features

User Roles (as shown in your diagram)

- **Students**: Browse, search, and book hostels
- **Hostel Owners**: Manage their hostel listings
- **Admins**: System administration and oversight

Key Functionality

1. **Authentication System**
   - Student login/signup with university credentials
   - Hostel owner registration and verification
   - Admin dashboard access
2. **Hostel Discovery**
   - Search by location, price, amenities
   - Map-based search using Maps API

    - Filters (price range, distance from campus, etc.)
3. **Booking System**
    - Room selection
    - Date selection
    - Paystack integration for mobile money payments
    - Booking confirmation and receipt
4. **Hostel Management (for owners)**
    - Add/edit hostel listings
    - Manage room availability
    - View bookings and occupancy
    - Process payments and refunds
5. **Admin Features**
    - Monitor system activity
    - Manage users (students and hostel owners)
    - Generate reports on bookings and usage
    - System configuration options

3. UI Design Elements

Homepage

- Hero section showcasing the best hostels
- Quick search feature
- Featured hostels near the university
- Testimonials from students

Navigation

- Main navigation bar with: Home, Hostels, Bookings, About, Contact
- User account dropdown (My Bookings, Profile, etc.)
- Footer with links to important university resources

Hostel Listings Page

- Grid/list view toggle
- Map view option
- Detailed filters
- Sort by price, rating, distance

Single Hostel Page

- Photo gallery
- Amenities list
- Room types and availability
- Reviews and ratings

- Location map
- Booking form

4. Database Schema (MongoDB)

```
// User Schema
{
  _id: ObjectId,
  role: String, // "student", "hostelOwner", "admin"
  name: String,
  email: String,
  password: String, // hashed
  studentId: String, // for students
  phoneNumber: String,
  profileImage: String,
  createdAt: Date,
  updatedAt: Date
}

// Hostel Schema
{
  _id: ObjectId,
  name: String,
  description: String,
  address: String,
  location: {
    type: "Point",
    coordinates: [Number, Number] // [longitude, latitude]
  },
  owner: ObjectId, // reference to User
  images: [String],
  amenities: [String],
  policies: String,
  verified: Boolean,
  rating: Number,
  reviews: [ObjectId], // reference to Reviews
  createdAt: Date,
  updatedAt: Date
}

// Room Schema
{
  _id: ObjectId,
  hostelId: ObjectId, // reference to Hostel
  name: String,
  type: String, // e.g., "single", "double", "shared"
  description: String,
  price: Number,
  capacity: Number,
  amenities: [String],
  images: [String],
  availability: Boolean,
  createdAt: Date,
  updatedAt: Date
}

// Booking Schema
{
  _id: ObjectId,
  roomId: ObjectId, // reference to Room
  hostelId: ObjectId, // reference to Hostel
  userId: ObjectId, // reference to User
  checkInDate: Date,
  checkOutDate: Date,
  status: String, // "pending", "confirmed", "cancelled"
  paymentStatus: String, // "pending", "paid", "refunded"
  paymentId: String, // Paystack reference
  totalAmount: Number,
  createdAt: Date,
  updatedAt: Date
}

// Review Schema
{
  _id: ObjectId,
  hostelId: ObjectId, // reference to Hostel
  userId: ObjectId, // reference to User
  rating: Number,
  comment: String,
  createdAt: Date,
  updatedAt: Date
}
```

5. API Routes Structure

```
/api/auth
  - /login
  - /signup
  - /logout
```

```
/api/hostels
  - GET: List all hostels with filtering
  - POST: Create new hostel (hostelOwner only)
  - /:id - GET: Single hostel details
  - /:id - PUT: Update hostel details
  - /:id - DELETE: Remove hostel listing

/api/rooms
  - GET: List rooms by hostel
  - POST: Add new room (hostelOwner only)
  - /:id - GET, PUT, DELETE

/api/bookings
  - GET: User's bookings
  - POST: Create new booking
  - /:id - GET, PUT, DELETE
  - /verify-payment

/api/admin
  - /stats
  - /users
  - /settings
```

6. Integration with Paystack

1. Setup Paystack account and get API keys
2. Implement payment flow:
    - Initialize transaction
    - Redirect to Paystack payment page

    - Handle callback and verification
    - Update booking status based on payment status

7. Maps Integration

1. Implement the Maps API for:
    - Showing hostel locations on a map
    - Distance calculation from university
    - Location-based search
    - Directions to hostels

8. Unique Features to Make It Stand Out

1. **Virtual Tours**: 360° views of rooms and facilities
2. **Room Comparison**: Side-by-side comparison of different rooms
3. **Roommate Finder**: Optional feature for students to find compatible roommates
4. **Maintenance Requests**: Allow students to report issues directly through the app
5. **Academic Calendar Integration**: Show important university dates
6. **Local Services Map**: Show nearby shops, restaurants, and services
7. **Flexible Booking Options**: Semester-long, monthly, or short-term stays
8. **Reward System**: Points for loyal users, reviews, or referrals

9. Development Phases

Phase 1: Foundation

- Authentication system
- Basic hostel listings
- Simple search functionality

Phase 2: Core Features

- Advanced search and filters
- Booking system
- Payment integration

Phase 3: Enhanced Features

- Maps integration
- Reviews and ratings
- Admin dashboard

Phase 4: Premium Features

- Virtual tours
- Roommate finder
- Maintenance requests